Cocoa Tips and Tricks

Session 116

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Cocoa Tips and Tricks

Tony Parker
Cocoa Frameworks Engineer

Who's There?

Just starting out	
Some Cocoa experience	
Cocoa expert	

Topics

- Zombies
- Debugging
- Overlay views
- Key-value observing and concurrency

Zombies

Pop Quiz:
What do the following have in common?

Crash in objc_msgSend

```
#0   0x7fff8320e4cc in objc_msgSend
#1   0x100000f2d in -[ZombieAppDelegate crashA:] at ZombieAppDelegate.m:81
...
```

Crash in Notification Send

```
#0     0x7fff8320e4d0 in objc_msgSend
#1     0x7fff82b98615 in _nsnote_callback
#2     0x7fff8898742a in __CFXNotificationPost
#3     0x7fff88986de8 in _CFXNotificationPostNotification
...
```

Crash in Autorelease Pool Pop

```
#0    0x7fff8320e4cc in objc_msgSend
#1    0x7fff88a48056 in _CFAutoreleasePoolPop
#2    0x7fff82b80f33 in -[NSAutoreleasePool drain]
#3    0x7fff81d3f722 in -[NSApplication run]
#4    0x7fff81d2df6a in NSApplicationMain
...
```

Crash in Delegate Call

```
#0    0x7fff8320f588 in objc_msgSend_vtable5
#1    0x7fff81f8345f in -[NSTextField textShouldBeginEditing:]
#2    0x7fff81e3f151 in -[NSTextView(NSSharing)
                          shouldChangeTextInRanges:replacementStrings:]
#3    0x7fff81f8268a in -[NSTextView insertText:replacementRange:]
#4    0x7fff81f7c9e9 in -[NSTextInputContext handleTSMEvent:]
...
```

Disappointed Users

Answer

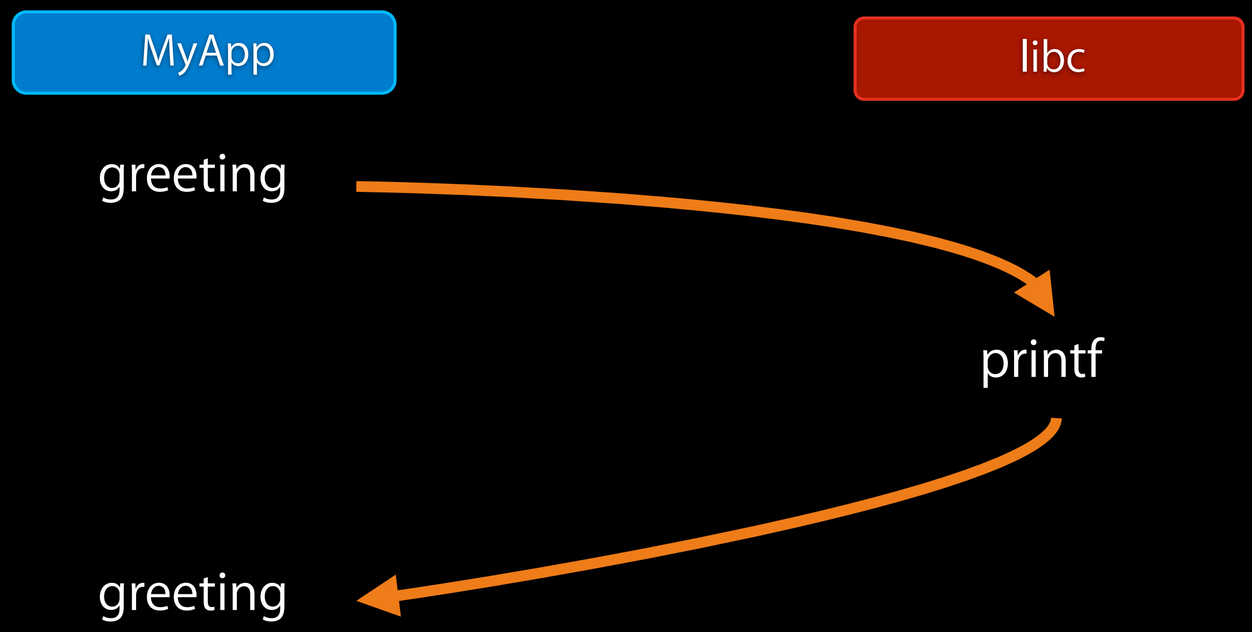
objc_msgS d is buggy!

Answer

- Result of accessing over-released memory
- But only the first one was in my code!

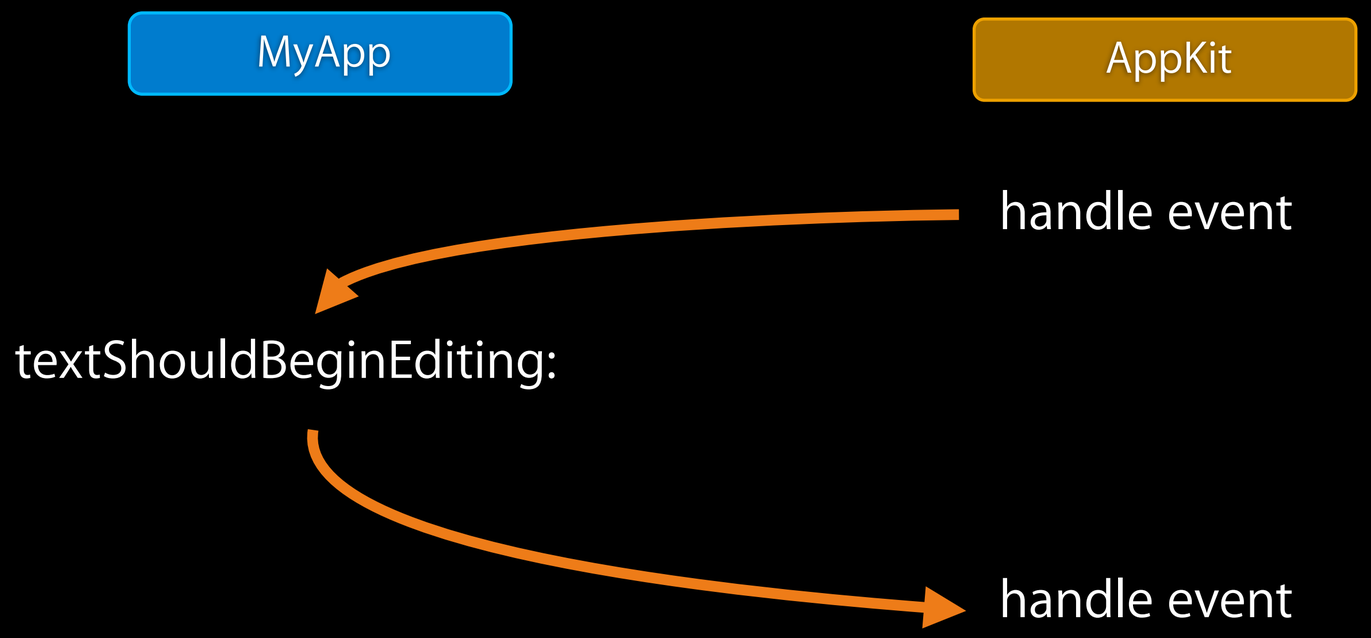
Libraries and Frameworks

```
void greeting() {
    printf("hello world\n");
}
```



Libraries and Frameworks

```
- (BOOL)textShouldBeginEditing:(NSText *)textObject {
    return timeIsRight;
}
```



“Don’t call us. We’ll call you.”

A Hollywood Agent

delegation, target-action, view drawing, responder chain and event handling, enumeration, archiving, notifications, key-value observing, documents, bindings, distributed objects, operations, run loops, undo, xml parsing, menu and toolbar validation, services, pasteboard, drag and drop, table and outline view data source, etc…

Zombies

What's a zombie?

- Class which stops execution of your program when a message is sent to a deallocated object
- Enabled by setting NSZombieEnabled environment variable to YES

How Do Zombies Work?

```
MyObj *obj = [[MyObj alloc] init];
```

1

```
[obj retain];
```

2

```
[obj release];
```

1

```
[obj release];
```

```
[obj dealloc];
```

x

Zombify

```
[obj doStuff];
```

Demo

Zombies

Summary

- Zombies help detect problems early
- Enable with NSZombieEnabled set to YES
- Use from Xcode or new Zombies instrument
- Note: Not used in garbage-collected applications

Fun Fact

- Bugs happen
- …even in Apple frameworks
- File a report at bugreport.apple.com
- Include:
  - System configuration
  - Regression info
  - Steps to reproduce
  - Test case
- Thank you!

Debugging

What You Already Know About gdb

- Controlling execution

- ‘print-object’ or ‘po’
  - Calls -debugDescription method on an object
    - …which calls -description
  - Implement -description on your own objects

gdb Tips

- Call other methods and functions
- Change state
- Execute alternate branches
- Run debugger commands automatically or conditionally

“Sometimes when I move shapes around in Sketch, they don’t draw right. Help! I have a school assignment due tomorrow!”

SketchKid2002

Demo

Debugging with gdb

“hay, wen i triz ta maeks ma lolcat picz in sketch, teh kitteh iz upsied down”

lolzmaekr

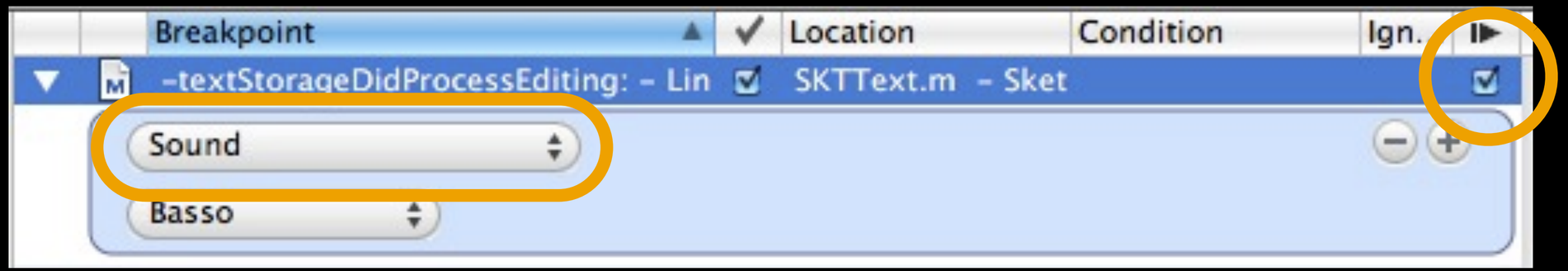
Demo

More debugging with gdb

Enable Breakpoints at Breakpoints

```
(gdb) br someEarlyFunc
Breakpoint 1 at ...
(gdb) br someLaterFunc
Breakpoint 2 at ...
(gdb) disable 2
(gdb) commands 1
>enable 2
>continue
>end
(gdb) commands 2
>disable 2
>end
(gdb)
```

Sound at a Breakpoint



Archive Object Graph

```
(gdb) call (BOOL)[NSKeyedArchiver archiveRootObject:foo
 toFile:@"/tmp/myfile.format"]

(gdb) set var foo=(id)[NSKeyedUnarchiver
 unarchiveObjectWithFile:@"/tmp/myfile.format"]
```

Caution

- gdb cannot stop all other threads when executing
- Methods which reenter might cause problems
- Watch out for deadlock
- Understand your application

Summary

- Call other methods and functions
- Change state
- Execute alternate branches
- Run debugger commands automatically or conditionally

dtrace Tips

- Low-level system debugging and profiling utility
- Can be used to investigate higher-level applications
- Via
  - Instruments
  - dtrace utility

Measuring Sketch Archiving Time

```
#!/usr/bin/env dtrace -qs
```

Measuring Sketch Archiving Time

```
#!/usr/bin/env dtrace -qs

objc$1:
{

}
```

Measuring Sketch Archiving Time

```
#!/usr/bin/env dtrace -qs

objc$1:SKTDocument:
{

}
```

Measuring Sketch Archiving Time

```
#!/usr/bin/env dtrace -qs

objc$1:SKTDocument:-dataOfType?error?:
{

}
```

Measuring Sketch Archiving Time

```
#!/usr/bin/env dtrace -qs

objc$1:SKTDocument:-dataOfType?error?:entry
{

}
```

Measuring Sketch Archiving Time

```
#!/usr/bin/env dtrace -qs

objc$1:SKTDocument:-dataOfType?error?:entry
{
    self->enterTime = vtimestamp;
}
```

Measuring Sketch Archiving Time

```
#!/usr/bin/env dtrace -qs

objc$1:SKTDocument:-dataOfType?error?:entry
{
    self->enterTime = vtimestamp;
}

objc$1:SKTDocument:-dataOfType?error?:return
{


}
```

Measuring Sketch Archiving Time

```
#!/usr/bin/env dtrace -qs

objc$1:SKTDocument:-dataOfType?error?:entry
{
    self->enterTime = vtimestamp;
}

objc$1:SKTDocument:-dataOfType?error?:return
{
    this->totalTime = vtimestamp - self->enterTime;

}
```

Measuring Sketch Archiving Time

```
#!/usr/bin/env dtrace -qs

objc$1:SKTDocument:-dataOfType?error?:entry
{
    self->enterTime = vtimestamp;
}

objc$1:SKTDocument:-dataOfType?error?:return
{
    this->totalTime = vtimestamp - self->enterTime;
    printf("%d\n", this->totalTime);
}
```

Demo

Measuring with dtrace

Stop Your App

```
#!/usr/bin/env dtrace -wqs

Cocoa_Autorelease$1:*:*:error_no_pool
{
    stop();
}
```

```
sudo ./nopool.d 4739

gdb attach 4739

Program received signal SIGSTOP, Stopped (signal).
0x00007fff88a483d5 in __NSAutoreleaseNoPool ()
(gdb)
```

Log Method Entry

```
objc$1:SKTDocument:*:entry
{
    printf("Entered: %s\n", probefunc);
}
```

```
Entered: -graphics
Entered: -observeUndoManagerCheckpoint:
Entered: -insertGraphics:atIndexes:
Entered: -startObservingGraphics:
Entered: -graphics
Entered: -graphics
Entered: -graphics
...
```

Debug Deadlock

```
objc$1:NSLock:-lock:entry
{


}

objc$1:NSLock:-lock:return
{


}
```

Debug Deadlock

```
objc$1:NSLock:-lock:entry
{
    self->lockObj = arg0;

}

objc$1:NSLock:-lock:return
{

    printf("%d locked %x\n", tid, self->lockObj);
}
```

Debug Deadlock

```
unsigned long lockOwner[unsigned long];

objc$1:NSLock:-lock:entry
{
    self->lockObj = arg0;

}

objc$1:NSLock:-lock:return
{
    lockOwner[self->lockObj] = tid;
    printf("%d locked %x\n", tid, self->lockObj);
}
```

Debug Deadlock

```
unsigned long lockOwner[unsigned long];

objc$1:NSLock:-lock:entry
{
    self->lockObj = arg0;
    printf("%d attempting to lock %x, owned by %d\n",
        tid, arg0, lockOwner[self->lockObj]);
}

objc$1:NSLock:-lock:return
{
    lockOwner[self->lockObj] = tid;
    printf("%d locked %x\n", tid, self->lockObj);
}
```

Debug Deadlock

```
objc$1:NSLock:-unlock:entry
{

}

objc$1:NSLock:-unlock:return
{

}
```

Debug Deadlock

```
objc$1:NSLock:-unlock:entry
{
    self->unlockObj = arg0;
}

objc$1:NSLock:-unlock:return
{
    lockOwner[self->unlockObj] = 0;
}
```

Debug Deadlock

```
objc$1:NSLock:-unlock:entry
{
    self->unlockObj = arg0;
}

objc$1:NSLock:-unlock:return
{
    lockOwner[self->unlockObj] = 0;
    printf("%d unlocked %x\n", tid, self->unlockObj);
}
```

Debug Deadlock

```
1 attempting to lock A, owned by 0
1 locked A
1 attempting to lock B, owned by 0
2 attempting to lock B, owned by 0
2 locked B
2 attempting to lock A, owned by 1
```

Debug Deadlock

```
1 attempting to lock A, owned by 0
1 locked A
1 attempting to lock B, owned by 0
2 attempting to lock B, owned by 0
2 locked B
2 attempting to lock A, owned by 1
```

Debug Deadlock

```
1 attempting to lock A, owned by 0
1 locked A
1 attempting to lock B, owned by 0
2 attempting to lock B, owned by 0
2 locked B
2 attempting to lock A, owned by 1
```

Debug Deadlock

```
1 attempting to lock A, owned by 0
1 locked A
1 attempting to lock B, owned by 0
2 attempting to lock B, owned by 0
2 locked B
2 attempting to lock A, owned by 1
```

Debug Deadlock

```
1 attempting to lock A, owned by 0
1 locked A
1 attempting to lock B, owned by 0
2 attempting to lock B, owned by 0
2 locked B
2 attempting to lock A, owned by 1
```

Root cause: Taking two locks in opposite order

Summary

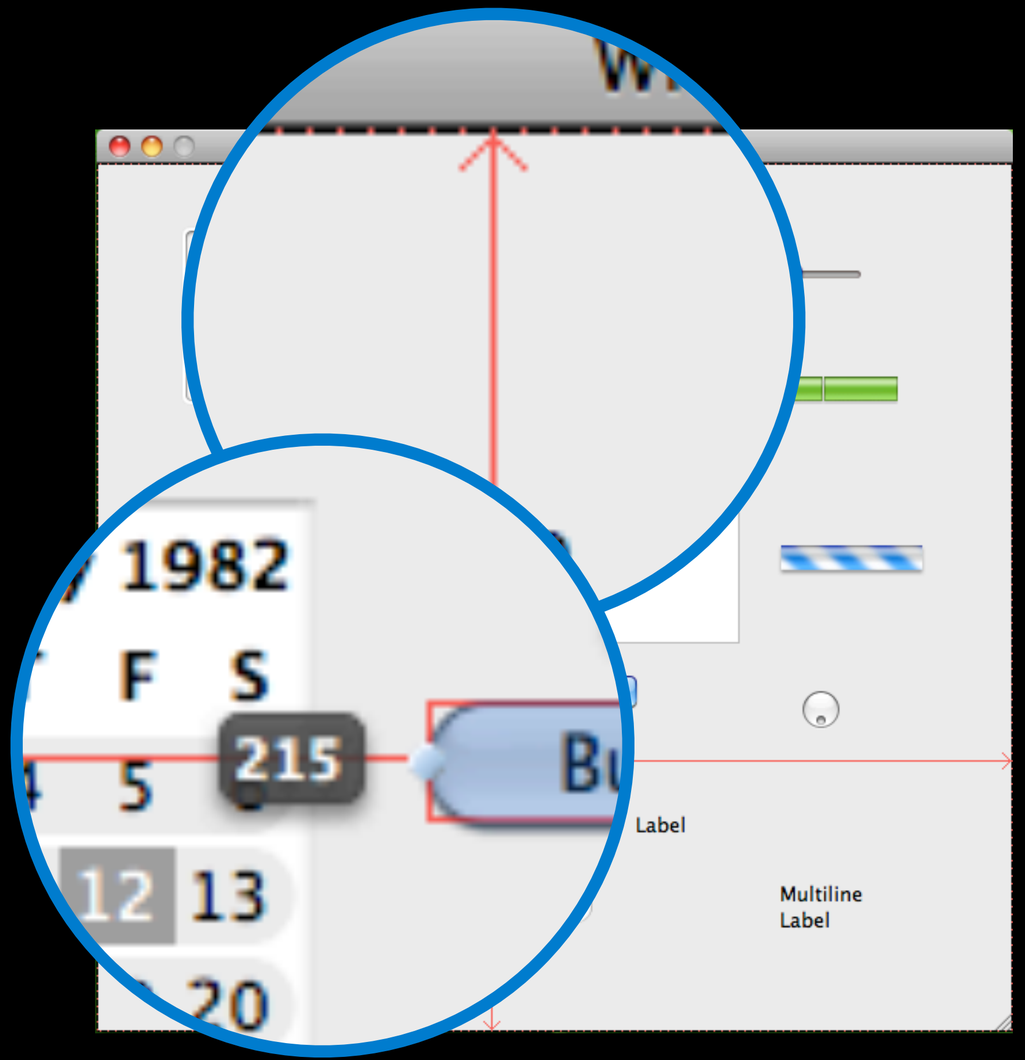
- Investigate code behavior and trends
- Answer questions about your application
- Add hooks almost anywhere in Cocoa with Obj-C provider

Fun Fact

- You can attach to a process that hasn’t launched yet with gdb

```
% gdb --waitfor TextEdit
Waiting for process 'TextEdit' to launch.
Attaching to process 3130.
(gdb)
```

Overlay Views


1982
F S
215
5
12 13
20
Label
Multiline
Label

Overlay Window

```
- (void)addChildWindow:(NSWindow *)childWin
               ordered:(NSWindowOrderingMode)place;
```

10.2

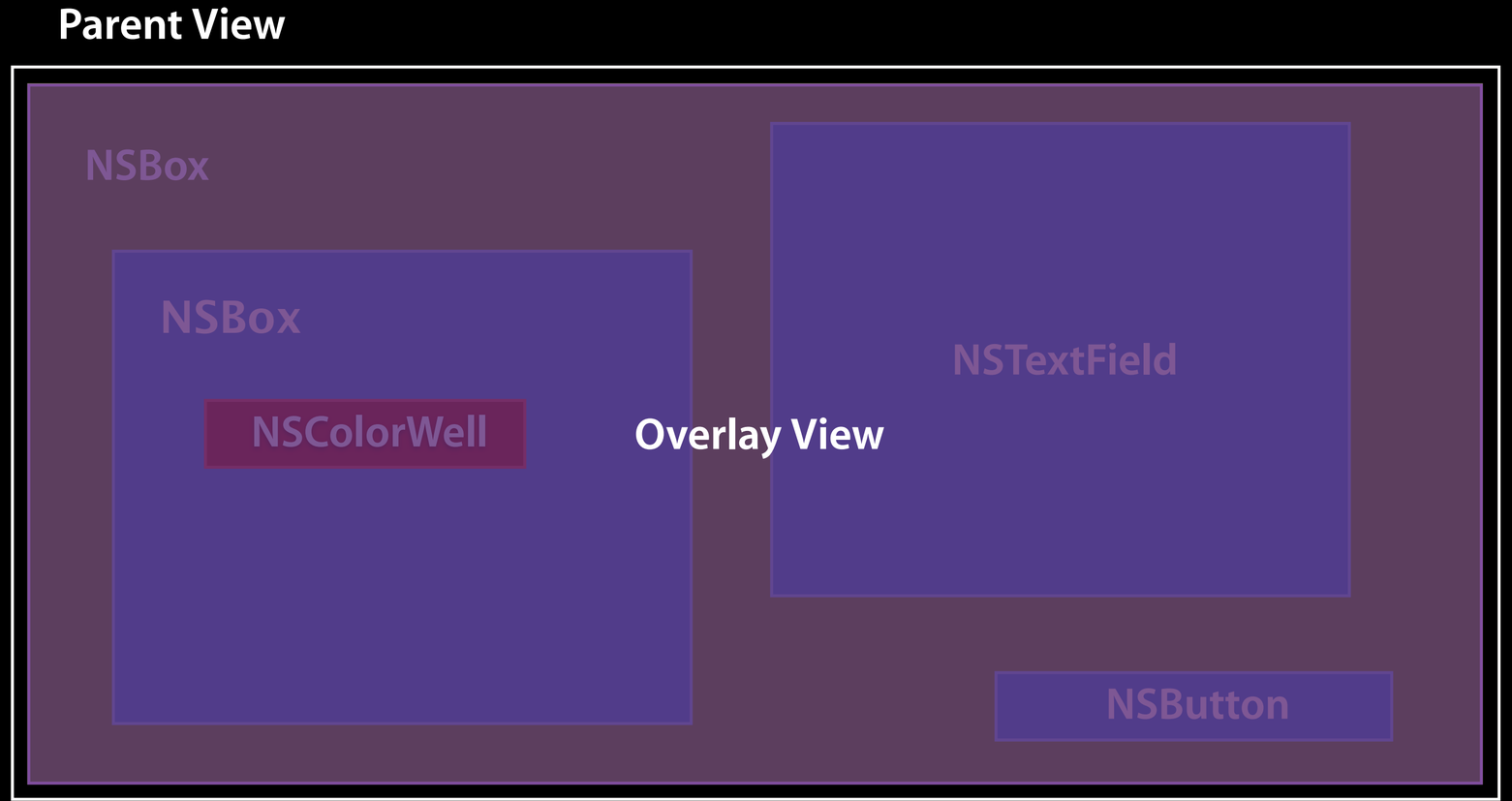
- Potentially more resource intensive
- Advantages
  - Can overlap NSOpenGLView or layer-backed views
  - Provide effect of drawing “outside” window

Presenting User Data with Table Views and Browsers	Nob Hill Thursday 10:30AM

Overlay View

- Sibling of other views
- Added as the last subview of parent
- Simple to implement, easy to understand

10.5


Parent View
NSBox
NSBox
NSColorWell
NSTextField
Overlay View
NSButton

Controller

```
- (void)awakeFromNib {




}
```

Controller

```
- (void)awakeFromNib {
    overlayView = [[OverlayView alloc] initWithFrame:
        [viewToOverlay frame]];
    [overlayView setAutoresizingMask:
        NSViewWidthSizable | NSViewHeightSizable];


}
```

Controller

```
- (void)awakeFromNib {
    overlayView = [[OverlayView alloc] initWithFrame:
        [viewToOverlay frame]];
    [overlayView setAutoresizingMask:
        NSViewWidthSizable | NSViewHeightSizable];

    [overlayView setHidden:!editing];
    [overlayView setOverlaidView:viewToOverlay];


}
```

Controller

```
- (void)awakeFromNib {
    overlayView = [[OverlayView alloc] initWithFrame:
        [viewToOverlay frame]];
    [overlayView setAutoresizingMask:
        NSViewWidthSizable | NSViewHeightSizable];

    [overlayView setHidden:!editing];
    [overlayView setOverlaidView:viewToOverlay];

    [[viewToOverlay superview] addSubview:overlayView];
}
```

Controller

```
- (void)setEditing:(BOOL)edit {




}
```

Controller

```
- (void)setEditing:(BOOL)edit {
    if (edit != [self editing]) {
        editing = edit;

    }
}
```

Controller

```
- (void)setEditing:(BOOL)edit {
    if (edit != [self editing]) {
        editing = edit;
        [overlayView setHidden:!editing];
    }
}
```

View

```
- (void)drawRect:(NSRect)rect;
```

View

```
- (void)drawRect:(NSRect)rect;

- (void)mouseMoved:(NSEvent *)theEvent;
- (void)mouseDown:(NSEvent *)theEvent;
- (void)mouseDragged:(NSEvent *)theEvent;
```

Demo

Overlay views

Summary

- Use to draw on top of overlapping sibling views
- Last subview of parent view
- Download “OverlayView” example

Fun Fact

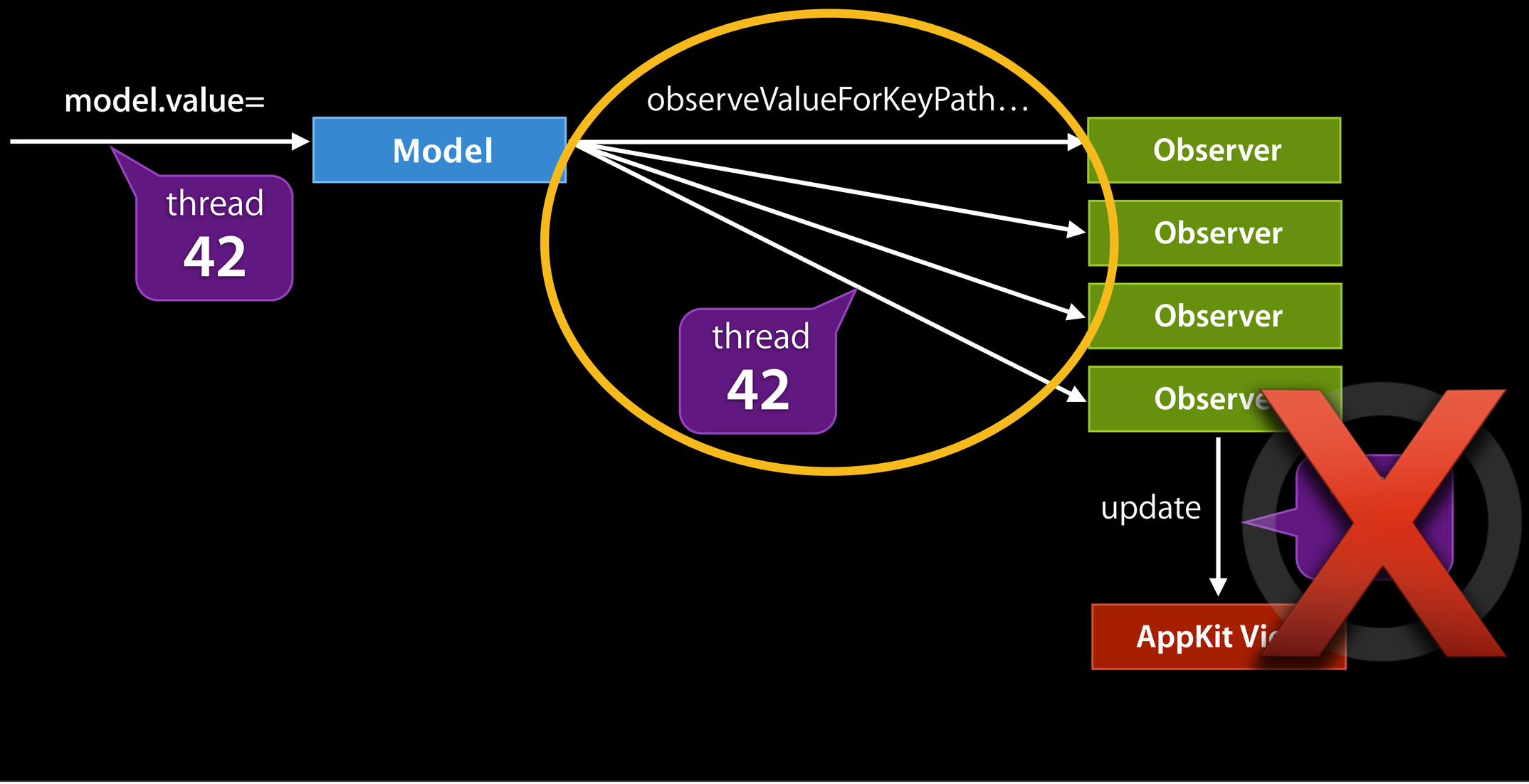
- When debugging, you can get a list of view information by calling a special description method

```
(gdb) po [aView _subtreeDescription]
[   A       ] h=--- v=--- NSView 0x3a55d0 f=(0,0,42,210) b=(-)…
[   AF      ] h=--& v=--& NSMatrix 0x3a5700 "1" f=(0,0,42,210)…

A=autoresizesSubviews, C=canDrawConcurrently, D=needsDisplay,
F=flipped, G=gstate, H=hidden (h=by ancestor), O=opaque,
P=preservesContentDuringLiveResize, S=scaled/rotated, W=wantsLayer
(w=ancestor wantsLayer), #=has surface
```

Key-Value Observing and Concurrency

KVO Overview

- Allows objects to be notified of changes to properties in other objects



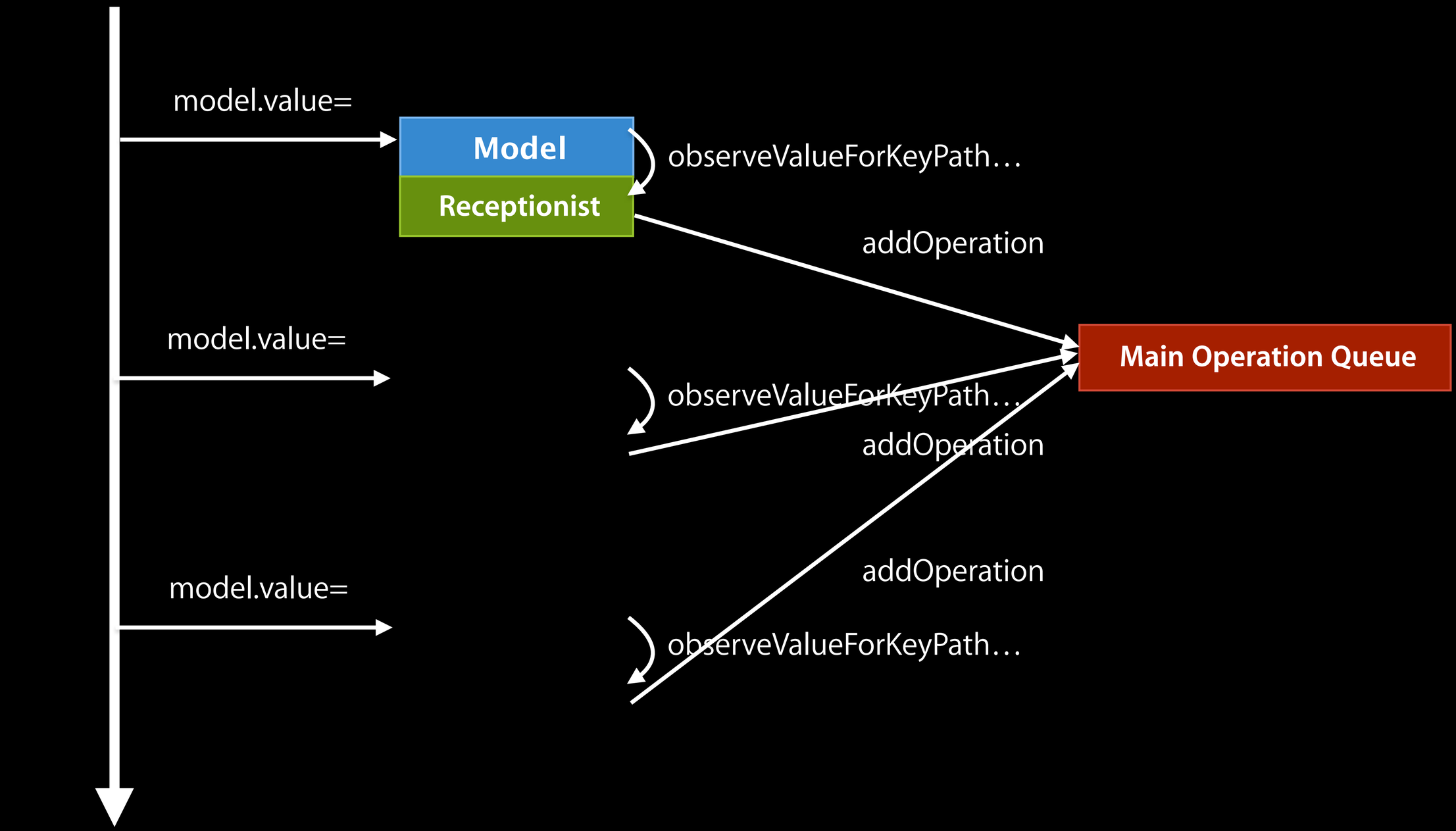
KVO Overview

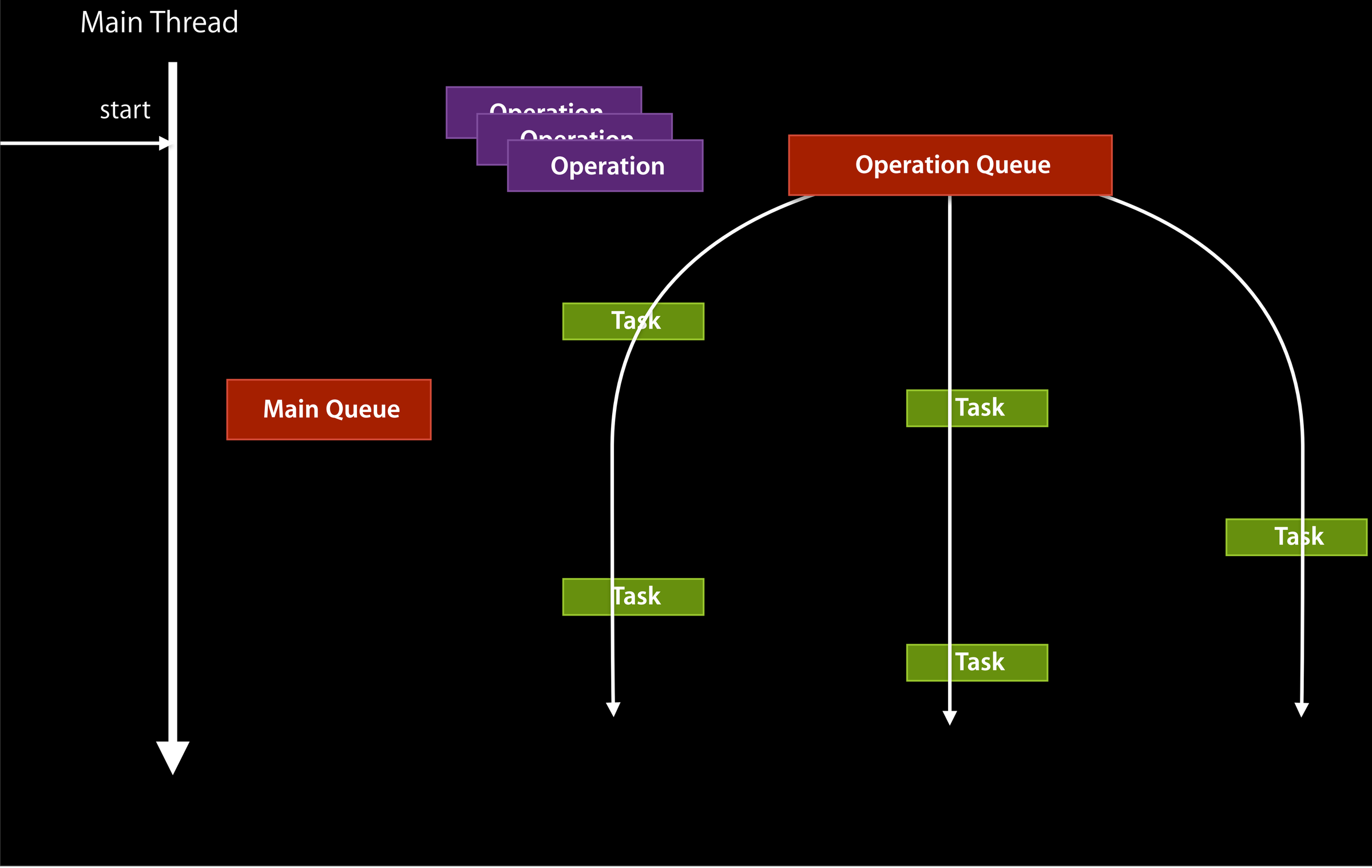
- How do I connect an update to a model object on a background thread to the user interface?

KVO Receptionist

- Thread middleman
- Receptionist will
  - Receive change notification
  - Execute task on another operation queue
- Use Snow Leopard block objects

10.6


Worker Thread
model.value=
Model
Receptionist
observeValueForKeyPath…
addOperation
Main Operation Queue
model.value=
observeValueForKeyPath…
addOperation
addOperation
model.value=
observeValueForKeyPath…


Main Thread
start
Operation
Operation
Operation
Operation Queue
Task
Main Queue
Task
Task
Task
Task

Receptionist

```
typedef void (^RCTaskBlock)(NSString *keyPath, id object,
NSDictionary *change);
```

Receptionist

```
typedef void (^RCTaskBlock)(NSString *keyPath, id object,
NSDictionary *change);

@interface RCReceptionist : NSObject {
    id observedObject;
    NSString *observedKeyPath;
    RCTaskBlock task;
    NSOperationQueue *queue;
}


@end
```

Receptionist

```
typedef void (^RCTaskBlock)(NSString *keyPath, id object,
NSDictionary *change);

@interface RCReceptionist : NSObject {
    id observedObject;
    NSString *observedKeyPath;
    RCTaskBlock task;
    NSOperationQueue *queue;
}
+ (id)receptionistForKeyPath:(NSString *)path object:(id)obj
queue:(NSOperationQueue *)queue task:(RCTaskBlock)task;
@end
```

Receptionist

```
+ (id)receptionistForKeyPath:(NSString *)path
                      object:(id)obj
                       queue:(NSOperationQueue *)queue
                        task:(RCTaskBlock)task {



}
```

Receptionist

```
+ (id)receptionistForKeyPath:(NSString *)path
                      object:(id)obj
                       queue:(NSOperationQueue *)queue
                        task:(RCTaskBlock)task {
    RCReceptionist *receptionist = [RCReceptionist new];


}
```

Receptionist

```
+ (id)receptionistForKeyPath:(NSString *)path
                      object:(id)obj
                       queue:(NSOperationQueue *)queue
                        task:(RCTaskBlock)task {
    RCReceptionist *receptionist = [RCReceptionist new];
    receptionist->task = [task copy];

}
```

Receptionist

```
+ (id)receptionistForKeyPath:(NSString *)path
                      object:(id)obj
                       queue:(NSOperationQueue *)queue
                        task:(RCTaskBlock)task {
    RCReceptionist *receptionist = [RCReceptionist new];
    receptionist->task = [task copy];
    receptionist->observedKeyPath = [path copy];
    receptionist->observedObject = [obj retain];
    receptionist->queue = [queue retain];



}
```

Receptionist

```
+ (id)receptionistForKeyPath:(NSString *)path
                      object:(id)obj
                       queue:(NSOperationQueue *)queue
                        task:(RCTaskBlock)task {
    RCReceptionist *receptionist = [RCReceptionist new];
    receptionist->task = [task copy];
    receptionist->observedKeyPath = [path copy];
    receptionist->observedObject = [obj retain];
    receptionist->queue = [queue retain];
    [obj addObserver:receptionist forKeyPath:path
        options:NSKeyValueObservingOptionNew |
        NSKeyValueObservingOptionOld context:0];

}
```

Receptionist

```
+ (id)receptionistForKeyPath:(NSString *)path
                      object:(id)obj
                       queue:(NSOperationQueue *)queue
                        task:(RCTaskBlock)task {
    RCReceptionist *receptionist = [RCReceptionist new];
    receptionist->task = [task copy];
    receptionist->observedKeyPath = [path copy];
    receptionist->observedObject = [obj retain];
    receptionist->queue = [queue retain];
    [obj addObserver:receptionist forKeyPath:path
        options:NSKeyValueObservingOptionNew |
        NSKeyValueObservingOptionOld context:0];
    return [receptionist autorelease];
}
```

Receptionist

```
- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary *)change
                       context:(void *)context {



}
```

Receptionist

```
- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary *)change
                       context:(void *)context {
    [queue addOperationWithBlock:^{

    }];
}
```

10.6

Receptionist

```
- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary *)change
                       context:(void *)context {
    [queue addOperationWithBlock:^{
        task(keyPath, object, change);
    }];
}
```

10.6

Controller

```
NSOperationQueue *mainQueue = [NSOperationQueue mainQueue];
```

10.6

Controller

```
NSOperationQueue *mainQueue = [NSOperationQueue mainQueue];
RCReceptionist *receptionist =
    [RCReceptionist receptionistForKeyPath:
                                    object:
                                     queue:
                                      task:

];
```

Controller

```
NSOperationQueue *mainQueue = [NSOperationQueue mainQueue];
RCReceptionist *receptionist =
    [RCReceptionist receptionistForKeyPath:@"value"
                                    object:model
                                     queue:mainQueue
                                      task:

];
```

Controller

```
NSOperationQueue *mainQueue = [NSOperationQueue mainQueue];
RCReceptionist *receptionist =
    [RCReceptionist receptionistForKeyPath:@"value"
                                    object:model
                                     queue:mainQueue
                                      task:
^(NSString *keyPath, id object, NSDictionary *change) {

}];
```

Controller

```
NSOperationQueue *mainQueue = [NSOperationQueue mainQueue];
RCReceptionist *receptionist =
    [RCReceptionist receptionistForKeyPath:@"value"
                                    object:model
                                     queue:mainQueue
                                      task:
^(NSString *keyPath, id object, NSDictionary *change) {
        // This will execute on the main queue
}];
```

Controller

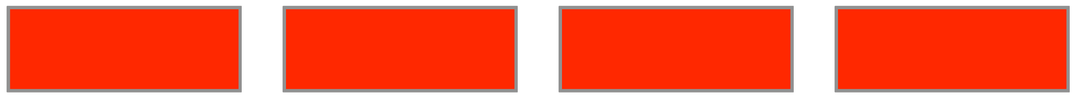
```
NSOperationQueue *mainQueue = [NSOperationQueue mainQueue];
RCReceptionist *receptionist =
    [RCReceptionist receptionistForKeyPath:@"value"
                                    object:model
                                     queue:mainQueue
                                      task:
     ^(NSString *keyPath, id object, NSDictionary *change) {
           // This will execute on the main queue
     }];


[workerQueue addOperationWithBlock:^{


}];
```

Controller

```
NSOperationQueue *mainQueue = [NSOperationQueue mainQueue];
RCReceptionist *receptionist =
    [RCReceptionist receptionistForKeyPath:@"value"
                                    object:model
                                     queue:mainQueue
                                      task:
     ^(NSString *keyPath, id object, NSDictionary *change) {
          // This will execute on the main queue
     }];


[workerQueue addOperationWithBlock:^{
    // This will execute on a background thread

}];
```

Controller

```
NSOperationQueue *mainQueue = [NSOperationQueue mainQueue];
RCReceptionist *receptionist =
    [RCReceptionist receptionistForKeyPath:@"value"
                                    object:model
                                     queue:mainQueue
                                      task:
     ^(NSString *keyPath, id object, NSDictionary *change) {
          // This will execute on the main queue
     }];

[receptionist retain];
[workerQueue addOperationWithBlock:^{
    // This will execute on a background thread
    [receptionist release];
}];
```



Demo

Key-value observing receptionist

Summary

- Moves notification to different queue, perhaps main queue
- Allows creator to define task
- Can be expanded to filter, redirect, or more
- Download “Receptionist” example

Tips and Tricks Recap

- Zombies
  - Detect memory management errors early
- Debugging
  - Find the root cause of an issue quickly
- Overlay views
  - Easily add overlapping view drawing
- KVO receptionist
  - Update model on background thread, but update UI on main thread

More Information

Matt Drance
Application Technologies Evangelist
mdrance@apple.com

Documentation
Zombies
http://developer.apple.com/qa/qa2004/qa1367.html

gdb
http://developer.apple.com/documentation/DeveloperTools/gdb/gdb/gdb_toc.html

Creating Custom Instruments with DTrace
http://developer.apple.com/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/CreatingaCustomInstrument/CreatingaCustomInstrument.html

More Information

More Documentation

Key-Value Observing
http://developer.apple.com/documentation/Cocoa/Conceptual/KeyValueObserving/KeyValueObserving.html

Concurrency Programming Guide
http://developer.apple.com/documentation/

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09